7.6 プログラミング言語

⌘ **機械語**
- CPU が理解しデコードできる命令からなる言語
- 各命令に 2 進符号が割り当てられている

⌘ **アセンブリ言語**
- 機械語と 1 対 1 に対応した命令からなる言語。ただし，各命令は人間が理解できるよう，ニーモニックで表現されている
- アセンブラが機械語プログラムに変換する

⌘ **高級言語**
- if 文，while 文など抽象度の高い構文を用いた言語
- 例えば，C 言語，C++ 言語，Java 言語など
- コンパイラが機械語プログラムに変換する

```
アセンブリ言語
   load    R0,    M[500]
   inc     R1,    R0
   store   M[501], R1

C 言語
   m[i+1] = m[i]++;
```

7.5 節の図のメモリのアドレス 100～102 番地に入っていた機械語プログラムに対応するアセンブリプログラムと C 言語プログラム

ただし，このアセンブリ言語は本文の説明用につくったもので，特定のアセンブリ言語の文法に従ったものではない。ここでは，配列 m[] の第 i 要素 m[i] がメモリの 500 番地 M[500] に対応するとしている

CPU に所望の動作をさせるためのプログラムは，**プログラミング言語**（programming language）を用いて記述するが，このプログラミング言語は，機械語，アセンブリ言語，高級言語に分類できる。

1) **機械語**（machine language）：CPU が直接理解し実行できるプログラミング言語で，2 進表現できる。各命令は，命令部，アドレス部，データ部から構成され，次のアセンブリ言語と 1 対 1 に対応している。

2) **アセンブリ言語**（assembly language）：人間が理解困難な各機械語命令に対して理解を助ける単語（ニーモニック，mnemonic）を割り当てた言語で，これで書かれた（アセンブリ）プログラムを機械語プログラムに変換するプログラムを**アセンブラ**（assembler）と呼ぶ。機械語に対応するため，命令の種類や記述方法は CPU に依存している。

3) **高級言語**：if 文，while 文など抽象度の高い構文を用いた言語で，CPU に依存せず標準化されている。これで書かれたプログラムを機械語プログラムに変換するプログラムを**コンパイラ**（compiler）と呼ぶ。

右上に示した C 言語の文は，配列 m の第 i 要素 m[i] のデータに 1 を加算し，その結果を第 i+1 要素 m[i+1] に代入するという操作であるが，コンパイラが変数 m[i] をメモリ中のアドレス 500 番地 M[500] と対応付けたとすると，この文に対応するアセンブリプログラムは，右上に示したものになり，その意味は下記である。

(1) 500 番地 M[500] にあるデータをレジスタ R0 に読み込む（load，ロード）。

(2) レジスタ R0 に 1 を加算（inc：increment，インクリメント）した結果をレジスタ R1 に入れる。

(3) レジスタ R1 のデータを 501 番地 M[501] に格納（store，ストア）する。

7.5 節の図のメモリのアドレス 100 ～ 102 番地に入っていた機械語プログラムは，このアセンブリプログラムに対応したものである。